

Flows in Temporal networks*

Eleni C. Akrida¹, Jurek Czyzowicz², Leszek Gąsieniec¹, Łukasz Kuszner³, and Paul G. Spirakis^{1,4}

¹Department of Computer Science, University of Liverpool, UK,
{Eleni.Akrida,L.A.Gasieniec,P.Spirakis}@liverpool.ac.uk

²Université du Québec en Outaouais, Dep. d'Informatique,
Gatineau, QC, Canada, jurek@uqo.ca

³Gdańsk University of Technology, Faculty of Electronics,
Telecommunications and Informatics, Poland,
kuszner@eti.pg.gda.pl

⁴CTI, Patras, Greece

June 6, 2016

Abstract

We introduce temporal flows on temporal networks, i.e., networks the links of which exist only at certain moments of time. Temporal networks were defined by Kempe et al. [27] (see also [33]). Our flow model is new and differs substantially from the “Flows over time” model, also called “dynamic flows” in the literature. We show that the problem of finding the maximum amount of flow that can pass from a source vertex s to a sink vertex t up to a given time is solvable in Polynomial time, even when node buffers are bounded. We then examine mainly the case of unbounded node buffers. We provide a (simplified) static *Time-Extended network*, which is of *polynomial size to the input* and whose static flow rates are equivalent to the respective temporal flow of the temporal network; via that, we prove that the maximum temporal flow is equal to the minimum *temporal s - t cut*. We further show that temporal flows can always be decomposed into flows, each of which is “pushed” only through a journey, i.e., a directed path whose successive edges have strictly increasing moments of existence. Using the latter, we provide *linear* expected time algorithms for the maximum s - t temporal flow problem in networks of bounded node degrees with uniform, random, unique availabilities of edges. We then consider *mixed* temporal networks, which have some edges with specified availabilities and some edges with random availabilities; we show that it is $\#P$ -complete to compute the *tails and expectations of the maximum temporal flow* (which is now a random variable) in a mixed

*This work was partially supported by (i) the School of EEE and CS and the NeST initiative of the University of Liverpool, (ii) the NSERC Discovery grant, (iii) the Polish National Science Center grant DEC-2011/02/A/ST6/00201, and (iv) the FET EU IP Project MULTIPLEX under contract No. 317532.

temporal network. Finally, we examine a Ford-Fulkerson inspired algorithm for maximum temporal flow in networks with a single availability for every edge and show that it computes the maximum temporal flow, i.e., there is an extension of the traditional algorithm [19] for our model. **Due to lack of space, the full paper with all proofs is attached as an appendix to be read at the discretion of the program committee.**

1 Introduction and motivation

1.1 Our model and the problem

It is generally accepted to describe a network topology using a graph, whose vertices represent the communicating entities and edges correspond to the communication opportunities between them. Consider a directed graph (network) $G(V, E)$ with a set V of n vertices (nodes) and a set E of m edges (links). Let $s, t \in V$ be two special vertices called the “source” and the “sink”, respectively. We assume that an infinite amount of a quantity, say, a liquid, is available in s at time zero. However, our network is *ephemeral*; each edge is available for use only at certain “days” in time, described by positive integers, and after some (finite) day in time, no edge becomes available again. For example, some edge $e = (u, v)$ may exist only at days 5 and 8; the reader may think of these days as instances of availability of that edge. Our “liquid”, located initially at node s , can flow in this ephemeral network through edges only at “days” at which the edges are available.

Each edge $e \in E$ in the network is also equipped with a *capacity* $c_e > 0$ which is a positive integer, unless otherwise specified. We also consider each node $v \in V$ to have an internal buffer (storage) $B(v)$ of maximum size B_v ; here, B_v is also a positive integer; initially, we shall consider both the case where $B_v = +\infty$, for all $v \in V$, and the case where all nodes have finite buffers. From Section 2 on, we only consider unbounded (infinite) buffers.

The *semantics* of the flow of our “liquid” within G are the following:

1. Let an amount x_v of liquid be at node v , i.e., in $B(v)$, at the “beginning” of day l , for some $l \in \mathbb{N}$. Let $e = (v, w)$ be an edge that exists at day l . Then, v may *push* a part of x_v through e at day l , as long as that part is at most c_e . This quantity will arrive to w at the “end” of *the same day*, l , and will be stored in $B(w)$.
2. At the end of day l , for any node w , some flows may arrive from edges (v, w) that were available at day l . Since each such quantity of liquid has to be stored in w , the sum of all flows incoming to w plus the amount of liquid that is already in w at the end of day l , i.e., after w has sent any flow out of it at the beginning of day l , must not exceed B_w .
3. Flow arriving at w at (the end of) day l can leave w only via edges existing at days $l' > l$.

Thus, our flows are not flow rates, but flow amounts (similar to considerations in *transshipment problems*).

We wish to provide here efficient solutions to the *Maximum temporal flow problem* (MTF): Given a directed graph G with edge availabilities, distinguished

nodes s, t , edge capacities and node buffers as previously described, and also given a specific “day” $l' > 0$, find the maximum value of the quantity of liquid that can arrive to t by (the end of) day l' .

Notice that no flow will arrive to t in fewer days than the “temporal distance of t from s ” (the smallest *arrival time* of any $s \rightarrow t$ path with strictly increasing days of availability on its consecutive edges; here, *arrival time* is the day of availability of the last edge on the path).

MTF is very different from the problem of standard (instantaneous) flows. Indeed, in a network with all node buffers and edge capacities being infinite, but *all edges existing only at the same day*, say $l = 5$, no flow can *ever* arrive to t .

Moreover, MTF is very different from the versions of (continuous or discrete) *dynamic flows* considered in [9, 17, 18, 22, 24], and references therein. Indeed, the “transit time” on each edge of our networks is *less than one day*, and only if the edge exists at that day. All units of flow that are located at the tail of an edge at a moment when the edge becomes available may pass through the edge virtually instantaneously, if the edge capacity allows it. That is, the “speed of travelling” via an edge at a day that it is available is virtually “infinite” only at that day and for any amount of flow at most the capacity of the edge. In fact, our model is an extreme case of a version of a discrete dynamic flows model called Dynamic Dynamic Flows [25, Chapter 8].

Also, in our model, the existence of node buffers (holdover flow) is *necessary*; *in contrast to all previous flow studies*, our networks cannot propagate flow without holdover flows, i.e., node buffers storing flow units.

So, we consider here ephemeral networks that change over time, as well as flows that are dynamic and the movement of which is determined by the temporal structure of the network.

1.2 Previous work

The traditional (static) network flows were extensively studied in the seminal book of Ford and Fulkerson [19] (see also Ahuja et al [1]) and the relevant literature is vast. *Dynamic network flows* (see, e.g., [25]) refer to *static* directed networks, the edges of which have capacities as well as transit times. Ford and Fulkerson [19] formulated and solved the dynamic maximum flow problem. For excellent surveys on dynamic network flows, the reader is also referred to the work of Aronson [6], the work of Powell [39], and the great survey by Skutella [41]. Dynamic network flows are also called *flows over time*. In [17], the authors review *continuous* flows over time where $f_e(\theta)$ is the rate of flow (per time unit) entering edge e at time θ ; the values of $f_e(\theta)$ are assumed to be Lebesgue-measurable functions. In our model, we assume that any flow amount that can pass through an edge at an instant of existence, will pass, i.e., our $f_e(\theta)$ is infinite in a sense. For various problems on flows over time, see [9, 17, 21, 22, 24, 26, 29, 30].

Classical static flows have recently been re-examined for the purpose of approximating their maximum value or improving their time complexity [5, 8, 15, 23, 31, 32, 37]. Network flows have also been used in multi-line addressing [14].

Temporal networks, defined by Kempe et al. [27], are graphs *the edges of which exist only at certain instants of time, called labels* (see also [33]). So,

they are a type of *dynamic* networks. Various aspects of temporal (and other dynamic) networks were also considered in [2, 3, 4, 7, 11, 12, 13, 16, 34, 35, 36, 40]. To the best of our knowledge, our work in this paper is the first to examine flows on temporal networks. Berman [10] proposed a similar model to temporal networks, called *scheduled networks*, in which each edge has separate departure and arrival times; he showed that the max-flow min-cut theorem holds in scheduled networks, when edges have *unit capacities*.

Perhaps the closest model to ours that has been examined in the flows literature is the “*Dynamic*¹ dynamic network flows”, studied in the PhD thesis of Hoppe [25, Chapter 8]. In [25, Chapter 8], Hoppe introduces *mortal edges* that exist between a start and an end time; still, Hoppe assumes finite speed of transmission on the edges. Thus, our model is an extreme case of the latter, since we assume that edges exist only at specific days (instances) and that transit rates are *virtually* infinite.

1.3 Our results

We introduce flows in Temporal Networks for the first time. We are interested in the maximum total amount of flow that can pass from s to t during the lifetime of the network; notice that the edges of the network exist only at some days during the lifetime, different in general for each edge.

In Section 1.4, we formulate the problem of computing the maximum temporal flow and show that it can be solved in polynomial time, even when the node capacities are finite (bounded holdover flow). This is in contrast to the NP-hardness result conjectured by Hoppe [25, personal communication with Klinz] for bounded holdover flows in dynamic dynamic networks.

The remainder of the paper concerns networks in which the nodes have unbounded buffers, i.e., buffers with infinite capacity. In Section 2.2, we define the corresponding time-extended network (*TEG*) which converts our problem to a static flow problem (following the time-extended network tradition in the literature [19]). However, we manage to simplify *TEG* into a *simplified time-extended network* (*STEG*), the size of which, i.e., number of nodes and edges, is *polynomial on the input*, and not exponential as usual in flows over time. Using the *STEG*, we prove that maximum s - t flows in temporal networks are equal to *minimum temporal s - t cuts*; temporal cuts extend the traditional cut notion, since the edges included in a cut need not exist at the same day(s) in time.

Since in our model flows have to follow *journeys*, i.e., directed paths whose time existence of successive edges strictly increases, we examine the issue of journey-decomposition of flows in temporal networks; we show in Section 2.3 that temporal flows are always decomposable into a set of flows, each moving through a particular journey. So, one can always find a permutation of journeys such that they select them one after the other, push as much flow as possible, remove the journey from the graph, and repeat, resulting in the maximum flow. However, the order of journeys in that decomposition affects the final flow value that is returned. We demonstrate cases where the (*expected*) number of journeys from s to t is constant; for example, in networks of bounded out-degree and random edge availabilities, we get *linear*-time maximum temporal flow algorithms as we show in Section 3.1.

¹The first “dynamic” term refers to the dynamic nature of the underlying graph, i.e., appearance and disappearance of its edges

We also introduce mixed temporal networks, in which the availabilities of some edges are random and the availabilities of some other edges are specified. In such networks, the value of the maximum temporal flow becomes a random variable; in Section 3.2 we show for mixed networks that it is $\#\mathbf{P}$ -complete to compute tails and expectations of the maximum temporal flow.

Finally, in Section 4 we examine a Ford-Fulkerson inspired algorithm for finding a maximum temporal flow in networks with a single availability for every edge and show its correctness, i.e., there is an extension of the traditional algorithm [19] for our model (and, thus, a way around the fact that the order of journeys in a flow decomposition affects the final flow value).

1.4 Formal Definitions

Definition 1 ((Directed) Temporal Graph). *Let $G = (V, E)$ be a directed graph. A (directed) temporal graph on G is an ordered triple $G(L) = (V, E, L)$, where $L = \{L_e \subseteq \mathbb{N} : e \in E\}$ assigns a finite set L_e of discrete labels to every edge (arc) e of G . L is called the labelling of G . The labels, L_e , of an edge $e \in E$ are the integer time instances (e.g., days) at which e is available.*

Definition 2 (Time edge). *Let $e = (u, v)$ be an edge of the underlying digraph of a temporal graph and consider a label $l \in L_e$. The ordered triplet (u, v, l) , also denoted as (e, l) , is called time edge. We denote the set of time edges of a temporal graph $G(L)$ by E_L .*

A basic assumption that we follow here is that when a (flow) entity passes through an available edge e at time t , then it can pass through a subsequent edge only at some time $t' \geq t + 1$ and only at a time at which that edge is available. In the tradition of assigning “transit times” in the dynamic flows literature, one may think that any edge e of the graph has some *transit time*, tt_e , with $0 < tt_e < 1$, but *otherwise arbitrary and not specified*. Henceforth, we will use $tt_e = 0.5$ for all edges e , without loss of generality in our results; any value of tt_e between 0 and 1 will lead to the same results in our paper.

Definition 3 (Journey). *A temporal path or journey j from a vertex u to a vertex v , $u \rightarrow v$ journey or (u, v) -journey, is a sequence of time edges (u, u_1, l_1) , (u_1, u_2, l_2) , \dots , (u_{k-1}, v, l_k) , such that $l_i < l_{i+1}$, for each $1 \leq i \leq k - 1$. We call the last time label, l_k , the arrival time of the journey.*

Definition 4 (Foremost journey). *A (u, v) -journey j in a temporal graph is called foremost journey if its arrival time is the minimum arrival time of all (u, v) -journeys’ arrival times, under the labels assigned to the underlying graph’s edges. We call this arrival time the temporal distance, $\delta(u, v)$, of v from u .*

Thus, no flow arrives to t (starting from s) on or before any time $l < \delta(s, t)$.

Definition 5 (Temporal Flow Network). *A temporal flow network $(G(L), s, t, c, B)$ is a temporal graph $G(L) = (V, E, L)$ equipped with:*

1. *a source vertex s and a sink (target) vertex t*
2. *for each edge e , a capacity $c_e > 0$; usually the capacities are assumed to be integers.*

3. for each node v , a buffer $B(v)$ of storage capacity $B_v > 0$; B_s and B_t are assumed to be infinite.

If all node capacities are infinite, we denote the temporal flow network by $(G(L), s, t, c)$.

Definition 6 (Temporal Flows in Temporal Flow Networks). Let $(G(L) = (V, E, L), s, t, c, B)$ be a temporal flow network. Let:

$$\begin{aligned}\delta_u^+ &= \{e \in E \mid \exists w \in V, e = (u, w)\} \\ \delta_u^- &= \{e \in E \mid \exists w \in V, e = (w, u)\}\end{aligned}$$

be the outgoing and incoming edges to u . Also, let $L_R(u)$ be the set of labels on all edges incident to u along with an extra label 0 (artificial label for initialization), i.e.,

$$L_R(u) = \bigcup_{e \in \delta_u^+ \cup \delta_u^-} L_e \cup \{0\}$$

A temporal flow on $G(L)$ consists of a non-negative real number $f(e, l)$ for each time-edge (e, l) , and real numbers $b_u^-(l), b_u^\mu(l), b_u^+(l)$ for each node $u \in V$ and each “day” l in $L_R(u)$. These numbers must satisfy all of the following:

1. $0 \leq f(e, l) \leq c_e$, for every time edge (e, l) ,
2. $0 \leq b_u^-(l) \leq B_u$, $0 \leq b_u^\mu(l) \leq B_u$, $0 \leq b_u^+(l) \leq B_u$, for every node u and every $l \in L_R(u)$
3. for every $e \in E$, $f(e, 0) = 0$,
4. for every $v \in V \setminus \{s\}$, $b_v^-(0) = b_v^\mu(0) = b_v^+(0) = 0$,
5. for every $e \in E$ and $l \notin L_e$, $f(e, l) = 0$,
6. at time 0 there is an infinite amount of flow “units” available at the source s ,
7. for every $v \in V \setminus \{s\}$ and for every $l \in L_R(v)$, $b_v^-(l) = b_v^+(l_{\text{prev}})$, where l_{prev} is the largest label in $L_R(v)$ that is smaller than l ,
8. (Flow out on day l) for every $v \in V \setminus \{s\}$, for every l , and for every outgoing edge $e = (v, w)$ of v , $b_v^\mu(l) = b_v^-(l) - \sum_e f(e, l)$,
9. (Flow in on day l) for every $v \in V \setminus \{s\}$, for every l , and for every incoming edge $e = (w, v)$ of v , $b_v^+(l) = b_v^\mu(l) + \sum_e f(e, l)$.

Note. One may think of $b_v^-(l), b_v^\mu(l), b_v^+(l)$ as the buffer content of liquid in v at the “morning”, “noon”, i.e., after the departures of flow from v , and “evening”, i.e., after the arrivals of flow to v , of day l .

Note. For a feasible temporal flow f on an acyclic $G(L)$, if one could guess the (real) numbers $f(e, l)$ for each time-edge (e, l) , then the numbers $b_v^-(l), b_v^\mu(l), b_v^+(l)$, for every $v \in V$, can be computed by a single pass of the topological order of the vertices of the (acyclic) $G(L)$ from s to t . This can be done by following (1) through (9) from Definition 6 from s to t .

Definition 7 (Value of a Temporal Flow). *The value $v(f)$ of a temporal flow f is $b_t^+(l_{max})$ under f , i.e., the amount of liquid that, via f , reaches t during the lifetime of the network (l_{max} is the maximum label in L). If $b_t^+(l_{max}) > 0$ for a particular flow f , we say that f is feasible.*

Problem (Maximum Temporal Flow (MTF)). *For a given temporal flow network $(G(L) = (V, E, L), s, t, c, B)$ and a given $d \in \mathbb{N}^*$, compute the maximum $b_t^+(d)$ over all flows f in the network.*

Note that if d is not a label in L , it is enough to compute the maximum $b_t^+(l_m)$ over all flows, where l_m is the maximum label in L that is smaller than d .

Henceforth, we assume that $d = l_{max}$ unless otherwise specified; notice that the analysis does not change: if $d < l_{max}$, one can remove all time-edges with labels larger than b and solve MTF in the resulting network with new maximum label at most d .

Note also that $b_t^+(l_{max})$ is not necessarily equal to the total outgoing flow from s during the lifetime of the network, where the lifetime is $l_{max} - l_{min}$, l_{min} being the smallest label in the network. For example, consider the network of Figure 1, where the labels of an edge are the numbers written next to it and its capacity is the number written inside the box; for $d = 5$, the *maximum flow by day 5* is $b_t^+(5) = 8$, i.e., the flow where 5 units follow the journey $s \rightarrow v \rightarrow t$ and 3 units follow the journey $s \rightarrow u \rightarrow t$; however, the total outgoing flow from t by day 5 is $10 > 8$.

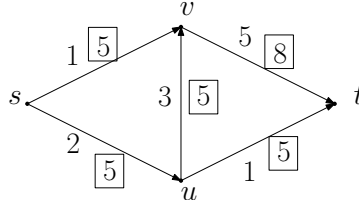


Figure 1: Outgoing flow from s is not always the same as maximum flow by some day d ; here $d = 5$.

Let Σ be the set of conditions of Definition 6. The optimization problem, Π :

$$\left\{ \begin{array}{ll} \text{maximize (over all } f) & b_t^+(d) \\ \text{under} & \Sigma \end{array} \right\}$$

is a *linear program* with unknown variables $\{f(e, l), b_v^-(l), b_v^+(l)\}$, $\forall l \in L, \forall v \in V$, since each condition in Σ is either a linear equation or a linear inequality in the unknown variables. Therefore, by noticing that the number of equations and inequalities are polynomial in the size of the input of Π , we get the following Lemma:

Lemma 1. *Maximum Temporal Flow is in P , i.e., can be solved in polynomial time in the size of the input, even when the node buffers are finite, i.e., bounded.*

Note. *If $n = |V|$, $m = |E|$ and $k = |E_L| = \sum_e |L_e|$, then MTF can be solved in sequential time polynomial in $n + m + k$ when the capacities and buffer sizes are polynomial in n . In the remainder of the paper, we shall investigate more efficient approaches for MTF.*

Note. Lemma 1 for bounded node buffers is in wide contrast with the claim that the corresponding problem in dynamic dynamic network flows is NP-complete [25, p. 82].

2 Temporal Networks with unbounded buffers at nodes

2.1 Basic remarks

We consider here the MTF problem for temporal networks with $B_v = +\infty$, $\forall v \in V$. Note that it is enough to only consider *acyclic* digraphs, since for every circulation (flow that moves in a journey C that is a cycle), we can have an equivalent -with respect to maximum flow at t - flow that *just waits* at a node until it can proceed to a simple journey towards t (see Fig. 2). Any flow moving into cycle C using increasing labels in the set L_2 must, in order to reach t , exit from v at a day in L_3 , greater than all the labels in L_2 . But then, that flow could “wait” in the buffer of v until that day and then exit.

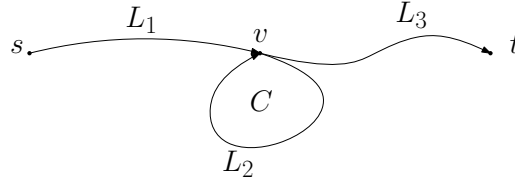


Figure 2: No need for cycles if node buffers are infinite.

Definition 8 (Temporal Cut). Let $(G(L) = (V, E, L), s, t, c)$ be a temporal flow network on an acyclic digraph G . A set of time-edges, S , is called a temporal cut (separating s and t) iff the removal of S from the network results in a temporal flow network with no journey from s to t .

Definition 9 (Minimal Temporal Cut). A set of time-edges, S , is called a minimal temporal cut (separating s and t) iff:

1. it is a temporal cut, and
2. the removal from the network of any $S' \subset S$ results in a temporal flow network with at least one journey from s to t , i.e., any proper subset of S is not a temporal cut.

Definition 10. Let S be a temporal cut of $(G(L) = (V, E, L), s, t, c)$. The capacity of the cut is $c(S) := \sum_{(e,l) \in S} c(e, l)$, where $c(e, l) = c_e$, $\forall l$.

In Figure 3, the numbers next to the edges are their availability labels and the numbers in the boxes are the edge capacities; here, a minimal temporal cut is $S = \{((s, v), 1), ((s, v), 7)\}$ with capacity $c(S) = 20$. Notice that another minimal cut is $S' = \{((v, t), 8)\}$ with capacity $c(S') = 2$.

It follows from the definition of a temporal cut:

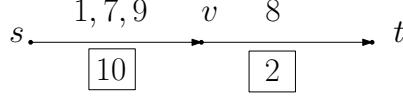


Figure 3: $S = \{((s, v), 1), ((s, v), 7)\}$ is a minimal cut.

Lemma 2. *Let S be a (minimal) temporal cut in $(G(L) = (V, E, L), s, t, c)$. If we remove S from $G(L)$, no flow can ever arrive to t during the lifetime of $G(L)$.*

Proof. The removal of S leaves no $s \rightarrow t$ journey and any flow from s needs at least one journey to reach t , by definition. \square

2.2 The time-extended flow network

Let $(G(L) = (V, E, L), s, t, c)$ be a temporal flow network on a directed acyclic graph G . Let E_L be the set of time edges of $G(L)$. Following the tradition in literature [19], we construct from $G(L)$ a *static* flow network called *time-extended* that corresponds to $G(L)$, denoted by $TEG(L) = (V^*, E^*)$. By construction, $TEG(L)$ admits the same maximum flow as $G(L)$. $TEG(L)$ is constructed as follows.

For every vertex $v \in V$ and for every time step $i = 0, 1, \dots, l_{max}$, we add to V^* a copy, v_i , of v . V^* also contains a copy of v for every time edge (x, v, l) of $G(L)$; in particular, we consider a copy v_{l+tt} of v in V^* , for some $l \in \mathbb{N}$, iff $(x, v, l) \in E_L$, for some $x \in V$. Notice that $0 < tt < 1$ (by definition of the transit times), so if a vertex $v \in V$ has an incoming edge e with label l and an outgoing edge with label $l + 1$, the copies v_{l+tt}, v_{l+1} of v in V^* will never be identical (see Figure 4).

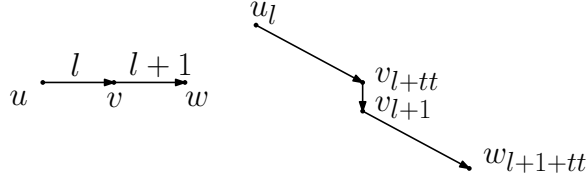


Figure 4: The copies of vertex v in $TEG(L)$.

E^* has a directed edge (called *vertical*) from a copy of vertex v to the *next* copy of v , for any $v \in V$. More specifically,

$$\forall v \in V, (v_i, v_j) \in E^* \iff \begin{cases} v_i, v_j \in V^*, & \text{and} \\ j > i, & \text{and} \\ \forall k > i : v_k \in V^* \implies k \geq j \end{cases}$$

Furthermore, for every time edge of $G(L)$, E^* has a directed edge (called *crossing*) as follows:

$$\forall u, v \in V, l \in \mathbb{N}, (u, v, l) \in E \iff (u_l, v_{l+tt}) \in E^*$$

Every crossing edge $e \in TEG(L)$ that connects copies of vertices $u, v \in V$ has the capacity of the edge $(u, v) \in G(L)$, $c_e = c_{u,v}$. Every vertical edge

$e \in TEG(L)$ has capacity $c_e = B_v = +\infty$. The source and target vertices in $TEG(L)$ are the first copy of s and the last copy of t in V^* , respectively. Note that $|V^*| \leq |V| \cdot l_{max} + |E_L|$ and $|E^*| \leq |V| \cdot l_{max} + 2|E_L|$.

We will now “simplify” $TEG(L)$ as follows: we convert vertical edges between consecutive copies of the same vertex into a *single vertical edge (with infinite capacity)* from the first to the last copy in the sequence and we remove all intermediate copies; we only perform this simplification when no intermediate node is an endpoint of a crossing edge. We call the resulting network *simplified time-extended* network and we denote it by $STEG(L) = (V', E')$.

In particular, for every vertex $v \in V$, V^* has a copy v_0 of v , and a copy for each time edge that includes v either as a first or as a last endpoint. We consider a copy v_l of v in V' iff $(v, x, l) \in E_L$, for some $x \in V$. We consider a copy v_{l+tt} of v in V^* iff $(x, v, l) \in E_L$, for some $x \in V$.

E' has a directed *vertical* edge from a copy of vertex v to the *next* copy of v , for any $v \in V$. More specifically,

$$\forall v \in V, (v_i, v_j) \in E' \iff \begin{cases} v_i, v_j \in V', & \text{and} \\ j > i, & \text{and} \\ \forall k > i : v_k \in V' \implies k \geq j \end{cases}$$

Furthermore, for every time edge of $G(L)$, we consider the *crossing* edge as in the time-extended graph, i.e.:

$$\forall u, v \in V, l \in \mathbb{N}, (u, v, l) \in E \iff (u_l, v_{l+tt}) \in E'$$

Every crossing edge $e \in STEG(L)$, i.e., every edge that connects copies of different vertices $u, v \in V$, has the capacity of the edge $(u, v) \in G(L)$, $c_e = c_{u,v}$. Every edge $e \in STEG(L)$ between copies of the same vertex $v \in V$ has capacity $c_e = B_v = +\infty$. The source and target vertices in $STEG(L)$ are the first copy of s and the last copy of t in V' respectively. Note that $|V'| \leq |V| + 2|E_L|$ and $|E'| \leq |V| + 3|E_L|$.

Denote the first copy of any vertex $v \in V$ in the time-extended network by v_{copy_0} , the second copy by v_{copy_1} , the third copy by v_{copy_2} , etc. Let also:

$$\begin{aligned} \delta_u^+ &= \{e \in E \mid \exists w \in V, e = (u, w)\} \\ \delta_u^- &= \{e \in E \mid \exists w \in V, e = (w, u)\} \end{aligned}$$

An $s \rightarrow t$ flow f in $G(L)$ defines an $s \rightarrow t$ flow (rate), f_R , in the time-extended network $STEG(L)$ as follows:

- The flow from the first copy of s to the next copy is the sum of all flow units that “leave” s in $G(L)$ throughout the time the network exists:

$$f(s_{copy_0}, s_{copy_1}) := \sum_{l \in \mathbb{N}} \sum_{e \in \delta_s^+} f(e, l)$$

- The flow from the first copy of any *other* vertex to the next copy is zero:

$$\forall v \in V \setminus s, f(v_{copy_0}, v_{copy_1}) := 0$$

- The flow on any crossing edge that connects some copy u_l of vertex $u \in V$ and the copy v_{l+tt} of some other vertex $v \in V$ is exactly the flow on the time edge (u, v, l) :

$$\forall (u_l, v_{l+tt}) \in E', f(u_l, v_{l+tt}) := f((u, v), l)$$

- The flow between two *consecutive* copies v_x and v_y , for some x, y , of the same vertex $v \in V$ corresponds to the units of flow stored in v from time x up to time y and is the difference between the flow *received* at the first copy through all incoming edges and the flow *sent* from the first copy through all outgoing crossing edges. So, $\forall v \in V, i = 1, 2, \dots$, it is:

$$f(v_{copy_i}, v_{copy_{i+1}}) := \sum_{z \in V'} f(z, v_{copy_i}) - \sum_{u \in V' \setminus v_{copy_{i+1}}} f(v_{copy_i}, u)$$

Example. Figure 5a shows a temporal network $G(L)$ with source s and sink t . The labels of an edge are shown next to the edge and the capacity of an edge is shown *written in a box* next to the edge. The respective simplified time-extended static graph $STEG(L)$ is shown in Figure 5b. The capacity of an edge is shown *written in a box* next to the edge. Notice that edges between copies of the source and edges between copies of the source have infinite capacities which are not shown in the figure.

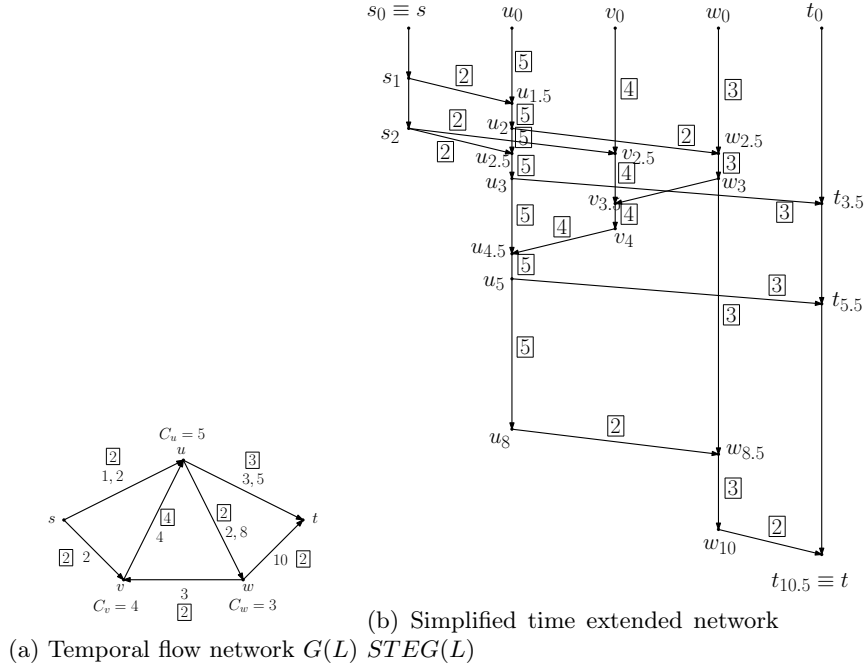


Figure 5: Constructing the Simplified time-extended network

Let f_R be a static flow rate in the static network $STEG(L)$ that corresponds to a temporal flow f in a temporal flow network $(G(L) = (V, E, L), s, t, c)$. By the construction of $STEG(L)$, it follows:

Lemma 3. *Given a temporal flow network $(G(L) = (V, E, L), s, t, c)$ on a directed acyclic graph G ,*

1. The maximum temporal flow (from s to t), $\max_{fv}(f)$, in $G(L)$ is equal to the maximum (standard) flow rate from s to t in the static network $STEG(L)$.
2. A temporal flow f is proper in $G(L)$ (i.e., satisfies all constraints) iff its corresponding static flow rate f_R is feasible in $STEG(L)$.

Lemma 4. *The minimum capacity s - t cut of the static network $TEG(L)$ is equal to the minimum capacity s - t cut of the static network $STEG(L)$.*

Proof. Any minimum capacity cut in either $TEG(L)$ or $STEG(L)$ uses crossing edges. But the crossing edges are the same in both networks. Therefore, the lemma holds. \square

We are now ready to prove the main Theorem of this section:

Theorem 1. *The maximum temporal flow in $(G(L) = (V, E, L), s, t, c)$ is equal to the minimum capacity (minimal) temporal cut.*

Proof. By Lemma 3, the maximum temporal flow in $G(L)$ is equal to the maximum flow rate from s to t in $TEG(L)$ and in $STEG(L)$. But in $STEG(L)$, the maximum s - t flow rate is equal to the minimum s - t cut [19]. Now, by Lemma 4, this cut is also equal in capacity to the minimum capacity s - t cut in $TEG(L)$. But any minimum capacity cut in $TEG(L)$ is only using crossing edges and thus corresponds to a temporal cut in $G(L)$, of the same capacity (since the removal of the respective time-edges leaves no $s \rightarrow t$ journey in $G(L)$). \square

It is also easy to see that:

Lemma 5. *Any static flow rate algorithm A that computes the maximum flow in a static, directed, acyclic, s - t network G of n vertices and m edges in time $T(n, m)$, also computes the maximum temporal flow in a $(G(L) = (V, E, L), s, t, c)$ temporal flow network in time $T(n', m')$, where $n' \leq n + 2|E_L|$ and $m' \leq n + 3|E_L|$.*

Proof. We run A on the static network $STEG(L)$ of n' vertices and m' edges. Note that $STEG(L)$ is, by construction, acyclic. \square

Note. *In contrast to all the dynamic flows literature, our simplified time-extended network has size (number of nodes and edges) linear on the input size of $G(L)$, and not exponential.*

2.3 Journeys and flow decomposition

We show here that any temporal flow from s to t (in temporal flow networks with unbounded node buffers) can be decomposed into temporal flows on some $s \rightarrow t$ journeys.

Lemma 6. *Let $(G(L) = (V, E, L), s, t, c)$ be a temporal flow network on a directed acyclic graph G . Let f be a temporal flow in $G(L)$ (f is given by the values of $f(e, l)$ for the time-edges (e, l) of $G(L)$). Then, there is a collection of $s \rightarrow t$ journeys j_1, j_2, \dots, j_k such that:*

1. $k \leq |E_L|$

2. $v(f) = v(f_1) + \dots v(f_k)$
3. f_i sends positive flow only on the time-edges of j_i

Proof. We show how to construct flows f_1, \dots, f_k (given f) with the above three properties. We do so via the following procedure:

```

1  $i := 1$ ;
2  $r := f$ , i.e.,  $\forall e, \forall l, r(e, l) := f(e, l)$ ;
3 while  $\sum_{l \in \mathbb{N}} \sum_{e \in \delta_s^+} r(e, l) > 0$  do
4   Find a maximal journey starting at  $t$  that uses only time-edges
      $(u, v, t)$  such that  $r((u, v), t) > 0$  and denote it by  $j_i$  /* A maximal
     journey is one that cannot be further “extended”, i.e.,
     we can’t find an outgoing time-edge from the last vertex
     with a larger label than the last label used in the
     journey */ Let  $f_{min}$  be the minimum of  $f_i((u, v), t)$  amongst the
     time-edges  $(u, v, t) \in j_i$ ;
5   For each  $(u, v, t) \in j_i$ , let  $f_i((u, v), t) := f_{min}$  and for all other
     time-edges  $(x, y, t^*)$ , let  $f_i((x, y), t^*) := 0$ ;
6   Let  $r((u, v), t) := r((u, v), t) - f_i((u, v), t)$ , for each  $(u, v)$  and  $t$ ;
7    $i := i + 1$ ;
```

The “residual” flow r is initialized to be equal to f , hence its value is the same as the value of f . At every step i , if the value of r is still positive, then we find a maximal journey j_i starting from s that is made entirely of time-edges with positive flow.

We define the flow f_i by sending along journey j_i the smallest of the amounts of flow sent by r along the time-edges of j_i . Note that f_i is a proper flow (i.e., satisfies the flow constraints) because for every time-edge we have $f_i((u, v), t) \leq r((u, v), t)$ and, by construction, we also have $r((u, v), t) \leq f((u, v), t)$; f was also a proper flow, so we have $f((u, v), t) \leq c_{uv}$. The same capacity constraints² are satisfied for the participating nodes in j_i . We then decrease $r((u, v), t)$ by $f_i((u, v), t)$ on each time-edge. This is still a proper flow, because we leave a non-negative flow on each time-edge.

After the update, the value of r decreases precisely by the same amount as the value of f_i , so we maintain that after i steps we have:

$$v(f) = v(r) + v(f_1) + \dots + v(f_i)$$

It remains to observe that after the update of r , at least one of the time-edges that had positive $r((u, v), t)$ has now $r((u, v), t) = 0$. This happens to the time-edge or time-edges that carry the minimum amount of flow along j_i . This means that after i steps, the number of time-edges (u, v, t) such that $r((u, v), t) > 0$ is at most $|E_L| - i$ and that, in particular, the algorithm finishes after at most $|E_L|$ iterations.

²The time-respecting constraint, i.e., the one that requires flow units to follow journeys, is clearly satisfied here, as well.

Let k be the number of iterations after which the algorithm finishes. At the completion of the algorithm, we have $\sum_{l \in \mathbb{N}} \sum_{e \in \delta_s^+} r(e, l) = 0$ and, thus, also $v(r) = 0$, hence we have:

$$v(f) = v(f_1) + \dots + v(f_i)$$

Therefore, the flows and journeys found by the algorithm satisfy all the requirements stated at the beginning. \square

Since Lemma 6 holds for any temporal flow, it also holds for the maximum temporal flow. Thus, one could, perhaps, find a maximum temporal flow by successively pushing as much flow quantity as possible via consecutive $s \rightarrow t$ journeys. However, the order of $s \rightarrow t$ journeys affects the flow value that eventually reaches t . Take as an example Figure 6; the maximum amount of flow that can arrive at t is via journey $s \rightarrow v \rightarrow t$ (pushing 7 units of flow to t) and then the journey $s \rightarrow u \rightarrow t$ (pushing another 5 units to t) for a total of 13 units of flow. However, if we first select journey $s \rightarrow u \rightarrow v \rightarrow t$, then the journey $s \rightarrow u \rightarrow t$ and then $s \rightarrow v \rightarrow t$, we can push 4 units of flow through the first, then 3 units of flow through the second (since the remaining capacity on (v, t) is now 3) and finally another 1 unit of flow through the last journey, giving a total of 8 units of flow reaching t .

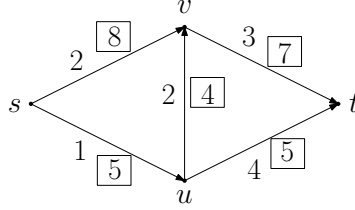


Figure 6: The order in which we push flow through $s \rightarrow t$ journeys matters.

Nonetheless, if the number of $s \rightarrow t$ journeys is *constant*, then we can get an efficient procedure for computing the maximum temporal flow as described in Algorithm 1.

Lemma 7. *Algorithm 1 returns the maximum temporal flow from s to t .*

Proof. The proof follows from Lemma 6: the maximum temporal flow corresponds to some journey decomposition; that decomposition will be captured by one of the permutations of all $s \rightarrow t$ journeys. \square

Corollary 1. *Let $(G(L) = (V, E, L), s, t, c)$ be a temporal flow network on a directed acyclic graph G , such that the number of $s \rightarrow t$ journeys is a fixed integer k . Then, Algorithm 1 finds the maximum flow from s to t during the lifetime of $G(L)$ in $O(k! |E_L|)$ time.*

3 Mixed Temporal Networks

In many practical cases, some edge availabilities are exactly specified, while some other edge availabilities are randomly chosen (due to security reasons, faults, etc.). The following definition is useful to describe such scenarios:

Algorithm 1: Successive Journeys algorithm

Input: A temporal flow network $(G(L) = (V, E, L), s, t, c)$

Output: A flow f arriving at t

```
1 Let all the different  $s \rightarrow t$  journeys be  $j_1, j_2, \dots, j_k$ , for some fixed integer  $k$ ;  
2 for each permutation  $\pi = (j_{\pi(1)}, j_{\pi(2)}, \dots, j_{\pi(k)})$  of  $j_1, j_2, \dots, j_k$  do  
3   For all time-edges  $(e, l)$  of  $G(L)$ , initialize  $\tilde{c}(e, l) = c(e, l)$ , where  
    $c(e, l) := c_e$  /*  $\tilde{c}(e, l)$  is the “remaining capacity” of the  
   time-edge  $(e, l)$  */  
4    $v(\pi) := 0$ , where  $v(\pi)$  is the (current) flow that arrives at  $t$   
   /* Initialize the flow that arrives at  $t$  to 0 */  
5   for each  $i = 1, 2, \dots, k$  do  
6     In the journey  $j_{\pi(i)}$  whose time-edges from  $s$  to  $t$ ,  
      $((e_1, l_1), \dots, (e_\lambda, l_\lambda))$ , for some  $\lambda \in \mathbb{N}^*$ , have remaining capacities  
      $\tilde{c}(e_1, l_1), \dots, \tilde{c}(e_\lambda, l_\lambda)$ , respectively, push from  $s$  to  $t$  an amount of  
     flow equal to the minimum of the remaining capacities, say  $\phi$ ;  
7     for each time edge  $(e, l)$  of  $j_{\pi(i)}$  do  
8        $\tilde{c}(e, l) := \tilde{c}(e, l) - \phi$  /* Note that after the end of the  
       loop, at least one time-edge of  $j_{\pi(i)}$  has zero  
       remaining capacity, and any subsequent journey that  
       uses such time-edges cannot carry any flow. */  
9      $v(\pi) = v(\pi) + \phi$  /* Add  $\phi$  to the flow that arrives at  $t$  */  
10 return  $\max_\pi v(\pi)$ 
```

Definition 11 (Mixed temporal networks). *Given a directed graph $G = (V, E)$ with a source s and a sink t in V , let $E = E_1 \cup E_2$, so that $E_1 \cap E_2 = \emptyset$, and:*

1. *the labels (availabilities) of edges in E_1 are specified, and*
2. *each of the labels of the edges in E_2 is drawn uniformly at random from the set $\{1, 2, \dots, \alpha\}$, independently of the others.*

We call such a network Mixed Temporal Network $[1, \alpha]$ and denote it by $G(E_1, E_2, \alpha)$.

Notice that the temporal networks that we studied in the previous sections are a special case of the mixed temporal networks, in which $E_2 = \emptyset$. However, with some edges being available at random times, the value of the maximum temporal flow (until time α) now becomes a random variable. In this section, we focus our attention to temporal networks that either have all their labels chosen uniformly at random, or are (fully) mixed. As in the previous section, we consider nodes with unbounded buffer capacity.

3.1 The journey decomposition of Temporal Networks with random availabilities

We will study here a special case of the mixed temporal networks $G(E_1, E_2, \alpha)$, where $E_1 = \emptyset$, i.e., *all* the edges in the network become available at random time instances.

Let $G = (V, E)$ be a directed acyclic graph of n vertices with one distinguished source, s , and one distinguished sink, t . Suppose that each edge $e \in E$ is available only at a *unique* moment in time (i.e., day) *selected uniformly at random from the set* $\{1, 2, \dots, \alpha\}$, *for some* $\alpha \in \mathbb{N}$, $\alpha > 1$; suppose also that the selections of the edges' labels are independent. Let us call such a network a Temporal Network with unique random availabilities of arcs, and denote it by $URTN(\alpha)$.

Lemma 8. *Let P_k be a directed $s \rightarrow t$ path of length k in G . Then, P_k becomes a journey in $URTN(\alpha)$ with probability at most $\frac{1}{k!}$.*

Proof. For a particular $s \rightarrow t$ path P_k of length k , let \mathcal{E} be the event that “ P_k is a journey”, \mathcal{D} be the event that “all k labels on P_k are different” and \mathcal{S} be the event that “at least 2 out of the k labels on P_k are equal”. Then, we have:

$$\begin{aligned} Pr[\mathcal{E}] &= Pr[\mathcal{E}|\mathcal{D}] \cdot Pr[\mathcal{D}] + Pr[\mathcal{E}|\mathcal{S}] \cdot Pr[\mathcal{S}] \\ &= Pr[\mathcal{E}|\mathcal{D}] \cdot Pr[\mathcal{D}] \\ &\leq Pr[\mathcal{E}|\mathcal{D}] \end{aligned}$$

Now, each particular *set* of k different labels in the edges of P_k is equiprobable. But for each such set, all permutations of the k labels are equiprobable and only one is a journey, i.e., has increasing order of labels. Therefore:

$$Pr[P_k \text{ is a journey}] \leq \frac{1}{k!}.$$

□

Assume now that the underlying graph G is such that, for each vertex v , the number of outgoing edges from v , $outdeg(v)$, is bounded by a constant d (bounded out-degree graph). Then, the number, $N(k)$, of the directed acyclic paths P_k of length k out of s are at most $N(k) \leq d^k$. So, the *expected number* of $s \rightarrow t$ journeys in the respective $URTN(\alpha)$ is:

$$\begin{aligned} \sum_{k=1}^n N(k) Pr[\text{each } P_k \text{ is a journey}] &\leq \sum_{k=1}^n d^k \frac{1}{k!} \\ &= O\left(\sum_{k=1}^n d^k \frac{1}{(\frac{k}{e})^k}\right) \\ &= O\left(\sum_{k=1}^n \left(\frac{de}{k}\right)^k\right) \\ &= O\left(\sum_{k=1}^{2de-1} \left(\frac{de}{k}\right)^k + \sum_{k=2de-1}^n \left(\frac{de}{k}\right)^k\right) \\ &\leq O((de)^{de}) + \sum_{\rho=1}^n \left(\frac{1}{2}\right)^\rho \\ &\leq O((de)^{de}) + 2, \end{aligned}$$

i.e., a (large) constant.

So, we can apply Algorithm 1 (Successive Journeys algorithm) to $URTN(\alpha)$ to calculate the maximum temporal flow in *expected time* $O(|E_L|)$, which is $O(|E|)$, since $URTN(\alpha)$ has one label per edge.

Remark 1. *This application has a large constant in the expected time and becomes practical for very large underlying graphs G .*

3.2 The complexity of computing the expected maximum temporal flow

We consider here the following problem:

Problem (Expected Maximum Temporal Flow). *What is the time complexity of computing the expected value of the maximum temporal flow, v , in $G(E_1, E_2, \alpha)$?*

Let us recall the definition of the class of functions $\#\mathbf{P}$:

Definition 12. [38, p.441] *Let Q be a polynomially balanced, polynomial-time decidable binary relation. The counting problem associated with Q is: Given x , how many y are there such that $(x, y) \in Q$? $\#\mathbf{P}$ is the class of all counting problems associated with polynomially balanced polynomial-time decidable functions.*

Loosely speaking, a problem is said to be $\#\mathbf{P}$ -complete if it is in $\#\mathbf{P}$ and a polynomial-time algorithm for it implies that $\#\mathbf{P} = \mathbf{FP}$, where \mathbf{FP} is the set of functions from $\{0, 1\}^*$ to $\{0, 1\}^*$ computable by a deterministic polynomial-time Turing machine³. For a more formal definition, see [38].

We now show the following:

Lemma 9. *Given an integer $C > 0$, it is $\#\mathbf{P}$ -complete to compute the probability that the maximum flow value v in $G(E_1, E_2, \alpha)$ is at most C , $\Pr[v \leq C]$.*

Proof. Recall that if $J = \{w_1, \dots, w_n\}$ is a set of n positive integer weights and we are given an integer $C \geq \sum_{i=1}^n \frac{w_i}{2}$, then the problem of computing the number, T , of subsets of J with total weight at most C is $\#\mathbf{P}$ -complete, because it is equivalent to counting the number of feasible solutions of the corresponding KNAPSACK instance [38].

Consider now the temporal flow network of Figure 7 where there are n directed disjoint two-edge paths from s to t . For the path with edges e_i, e'_i , via vertex v_i , the capacity of e_i is w_i and the capacity of e'_i is $w'_i \geq w_i$. In this network, $E_1 = \emptyset$ and $E_2 = E$, i.e., the availabilities of every edge are chosen independently and uniformly at random from $\{1, \dots, \alpha\}$. Also, assume that each edge selects a *unique* random label.

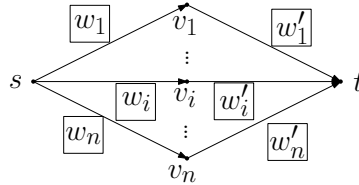


Figure 7: The network structure we consider

Clearly, the value of the maximum temporal flow from s to t until time $\alpha + tt$ is the sum of n random variables Y_i , $i = 1, \dots, n$, where Y_i is the value of the

³ $\{0, 1\}^* = \cup_{n \geq 0} \{0, 1\}^n$, where $\{0, 1\}^n$ is the set of all strings (of bits 0, 1) of length n

flow through the i^{th} path. Y_i is, then, w_i with probability $p_i = \frac{1}{2} - \frac{1}{\alpha}$, which is equal to the probability that the label l_{e_i} is smaller than the label $l_{e'_i}$, so that the path (e_i, e'_i) is a journey, and is zero otherwise. Then, $v = Y_1 + \dots + Y_n$ and it holds that $Pr[v \leq C] = Pr[\sum_{i=1}^n Y_i \leq C]$.

From now on, let J_k represent the set of all subsets of J of k weights. Let $\vec{g} = (g_1, \dots, g_n)$ be a specific assignment of weights to Y_1, \dots, Y_n , respectively, i.e., $g_i = w_i$ with probability $\frac{1}{2} - \frac{1}{\alpha}$ and, otherwise, $g_i = 0$. Denote by \tilde{g} the corresponding set of weights; so, $\tilde{g} \in J_k$. Then,

$$\begin{aligned} Pr[v \leq C] &= Pr[\sum_{i=1}^n Y_i \leq C] \\ &= \sum_{\vec{g}} Pr[Y_i = g_i, \forall i = 1, \dots, n] \cdot x(\vec{g}), \end{aligned} \quad (1)$$

where:

$$x(\vec{g}) = \begin{cases} 1 & , \text{ if } \sum_{i=1}^n g_i \leq C \\ 0 & , \text{ otherwise.} \end{cases}$$

For each particular \tilde{g} in J_k , the probability that it occurs is $(\frac{1}{2} - \frac{1}{\alpha})^k (\frac{1}{2} + \frac{1}{\alpha})^{n-k}$. So, from Equation 1 we get:

$$\begin{aligned} Pr[v \leq C] &= \sum_{k=0}^n \sum_{\tilde{g} \in J_k} x(\tilde{g}) \left(\frac{1}{2} - \frac{1}{\alpha}\right)^k \left(\frac{1}{2} + \frac{1}{\alpha}\right)^{n-k} \\ &= \left(\frac{1}{2} + \frac{1}{\alpha}\right)^n \sum_{k=0}^n \sum_{\tilde{g} \in J_k} x(\tilde{g}) \left(\frac{\frac{1}{2} - \frac{1}{\alpha}}{\frac{1}{2} + \frac{1}{\alpha}}\right)^k \end{aligned} \quad (2)$$

The following holds (using Bernoulli's inequality):

$$1 \geq \left(\frac{\frac{1}{2} - \frac{1}{\alpha}}{\frac{1}{2} + \frac{1}{\alpha}}\right)^k \geq \left(\frac{\frac{1}{2} - \frac{1}{\alpha}}{\frac{1}{2} + \frac{1}{\alpha}}\right)^n = \left(\frac{\alpha - 2}{\alpha + 2}\right)^n = \left(1 - \frac{4}{\alpha + 2}\right)^n \geq 1 - \frac{4n}{\alpha + 2} \quad (3)$$

Let T be $T = \sum_{k=0}^n \sum_{\tilde{g} \in J_k} x(\tilde{g})$. Then, we get from Equation 2 and Relation 3:

$$\begin{aligned} \left(\frac{1}{2} + \frac{1}{\alpha}\right)^n \left(1 - \frac{4n}{\alpha + 2}\right) T &\leq Pr[v \leq C] \leq \left(\frac{1}{2} + \frac{1}{\alpha}\right)^n T \Leftrightarrow \\ \left(1 - \frac{4n}{\alpha + 2}\right) T &\leq Pr[v \leq C] \frac{1}{\left(\frac{1}{2} + \frac{1}{\alpha}\right)^n} \leq T \Leftrightarrow \\ T - \frac{4nT}{\alpha + 2} &\leq \frac{Pr[v \leq C]}{\left(\frac{1}{2} + \frac{1}{\alpha}\right)^n} \leq T \end{aligned}$$

Now, assume that $\alpha + 2 > 4nT$ (note that T is exactly the number of subsets of J with total weight at most C); we can guarantee that by selecting α to be, for example, 2^n , or larger. Then, $0 < \frac{4nT}{\alpha + 2} < 1$. Let $\varepsilon = \frac{4nT}{\alpha + 2}$. Then, we get:

$$T - \varepsilon \leq \frac{Pr[v \leq C]}{\left(\frac{1}{2} + \frac{1}{\alpha}\right)^n} \leq T$$

Note that $\left(\frac{1}{2} + \frac{1}{\alpha}\right)^n$ can be represented by a polynomial in n number of bits and can be computed in polynomial time.

If we had a polynomial-time algorithm, A , to exactly compute $\Pr[v \leq C]$ for any C and α , then we could exactly compute (also in polynomial time) a number between $T - \varepsilon$ and T , for $0 < \varepsilon < 1$. But, this determines T exactly. So, such an algorithm A would solve a $\#\mathbf{P}$ -complete problem in polynomial time. \square

Remark 2. If each of the Y_i s was of the form $Y_i = w_i$ with probability $p_i = \frac{1}{2}$, and zero otherwise, then the reduction to the KNAPSACK problem would be immediate [20, 28]. However, the possibility of ties in the various l_{e_i} and $l_{e'_i}$ s excludes the respective journeys and the reduction does not carry out immediately.

Now, given a mixed temporal network $G(E_1, E_2, \alpha)$, let v be the random variable representing the maximum temporal flow in G .

Definition 13. The truncated by B expected maximum temporal flow of $G(E_1, E_2, \alpha)$, denoted by $E[v, B]$, is defined as:

$$E[v, B] = \sum_{i=1}^B \Pr[v = i]$$

Clearly, it is $E[v] = E[v, +\infty]$.

We are now ready to prove the main theorem of this section:

Theorem 2. It is $\#\mathbf{P}$ -complete to compute the expected maximum truncated Temporal Flow in a Mixed Temporal Network $G(E_1, E_2, \alpha)$.

Proof. Consider the single-labelled mixed temporal network $G(E_1, E_2, \alpha)$ of Figure 8, in which s has n outgoing disjoint directed paths of two edges e_i, e'_i to a node t_1 , and then there is an edge from t_1 to t . The capacity of each edge (s, v_i) , $i = 1, \dots, n$, is w_i , the capacity of each edge (v_i, t_1) , $i = 1, \dots, n$, is $w'_i \geq w_i$, and the capacity of the edge (t_1, t) is an integer B such that $\frac{1}{2} \sum_{i=1}^n w_i < B < \sum_{i=1}^n w_i$. The unique label of edge (t_1, t) is some $b \in \mathbb{N}$, $b > \alpha$, where α is the maximum possible label that the other edges may select; in particular, each of the edges $(s, v_i), (v_i, t_1)$, $i = 1, \dots, n$ receives a unique random label drawn uniformly and independently from $\{1, \dots, \alpha\}$.

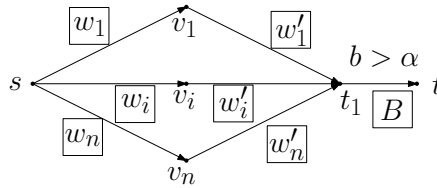


Figure 8: A $G(E_1, E_2, \alpha)$ where $E_1 = \{(t_1, t)\}$ with $l_{(t_1, t)} = b > \alpha$.

Clearly, the maximum temporal flow from s to t until time b is $v' = B$, if $v = \sum_{i=1}^n Y_i > B$, and is $v' = v = \sum_{i=1}^n Y_i$, otherwise; here Y_i , $i = 1, \dots, n$, is the random variable representing the flow passing from t to t_1 via v_i in the time until α .

So, if $E[v']$ is the expected value of v' , we have:

$$\begin{aligned} E[v'] &= \sum_{i=0}^B Pr[v = i] + B \cdot Pr[v > B] \\ &= E[v, B] + B(1 - Pr[v \leq B]) \end{aligned} \quad (4)$$

So, if we had a polynomial-time algorithm that could compute truncated expected maximum temporal flow values in mixed temporal networks, then we could compute $E[v']$ and $E[v, B]$; we could then solve Equation 4 for $Pr[v \leq B]$ and, thus, compute it in polynomial time. But to compute $Pr[v \leq B]$ is $\#\mathbf{P}$ -complete by Lemma 9. \square

4 A Ford-Fulkerson inspired technique for unique edge labels

In this section, we examine whether the traditional method of Ford and Fulkerson [19] for finding a maximum flow (rate) in static networks can be extended in our model of temporal networks; indeed, such a technique would overcome the difficulty that arises when deciding the order in which we should choose the $s \rightarrow t$ journeys for the journey decomposition of the maximum flow (see Section 2.3).

In this section, we focus our attention to the case of temporal networks in which each edge of the underlying graph receives a unique label of availability (single-labelled model) and the node capacities are infinite. Henceforth, we denote by l_e the (unique) time at which some edge $e \in E$ is available.

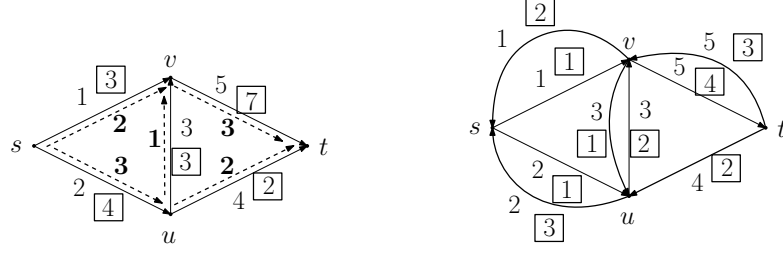
Definition 14 (Residual temporal network). *Given a single-labelled temporal flow network $(G(L) = (V, E, L), s, t, c)$ and a flow f in it, the residual temporal graph $G_f = (V', E', L')$ of $G(L)$ with respect to f is as follows:*

- $V' = V$
- **(Forward edges)** For each edge $e \in E$ with $f(e, l_e) < c_e$, we have $e \in E'$ with capacity $c_e - f(e, l_e)$ and time label l_e
- **(Backward edges)** For each edge $e = (u, v) \in E$ with $f(e, l_e) > 0$, we have $(v, u) \in E'$ with capacity $f(e, l_e)$ and time label l_e

Example. Figure 9a shows a temporal flow network $(G(L), s, t, c)$ and a flow f in it (dashed lines). Figure 9b shows the residual graph G_f of $G(L)$ with respect to f ; notice that the backward edges always have the same availability label as the respective (forward) edge of $G(L)$.

Algorithm 2 is the main algorithm of this section and describes how we can use the residual temporal network to augment the value of the flow in $G(L)$.

Notice that, by construction, at each iteration of Algorithm 2 it holds for the respective f that $v(f) = \sum_{l \in \mathbb{N}} \sum_{e \in \delta_s^+} f(e, l)$, i.e., at each iteration all the flow units that leave s arrive at t over time. Notice, also, that we only consider flow(s) on the edges of $G(L)$ and not on the residual temporal network; for every flow that can be constructed in $G(L)$ we have a respective temporal network (with no flow in it). The following lemmas are towards proving the correctness of Algorithm 2.



(a) Temporal flow network $(G(L), s, t, c)$ and flow f (b) Residual temporal network G_f

Figure 9: Constructing the residual temporal network given a flow in the temporal flow network

Algorithm 2: Temporal Ford-Fulkerson algorithm

Input: A temporal flow network $(G(L) = (V, E, L), s, t, c)$
Output: A maximum flow f on $G(L)$

```

1 for every edge  $e \in E$  do
2    $f(e, l_e) = 0$ ;
3  $G_f$  is the residual temporal network of  $G(L)$  with respect to  $f$ ;
4 while there exists an  $s \rightarrow t$  journey in  $G_f$  do
5   Let  $J$  be a  $s \rightarrow t$  journey in  $G_f$ ;
6    $f := \text{augment}(f, J)$ ;
7   Construct new residual graph  $G_f$  /* The construction of  $G_f$  is
   according to Definition 14 for the current flow  $f$  on  $G(L)$ 
   */
8 Function  $\text{Augment}(f, J)$ 
9   Let  $b$  be the minimum capacity of the edges in  $J$ ;
10  for every edge  $(u, v)$  in  $J$  do
11    if  $(u, v)$  is a forward edge of  $G_f$  then
12       $f((u, v), l_{(u, v)}) := f((u, v), l_{(u, v)}) + b$  /* Increase the flow
      on the edge  $(u, v)$  in  $G(L)$  */
13    else
14       $f((v, u), l_{(v, u)}) := f((v, u), l_{(v, u)}) - b$  /* Decrease the flow
      on the edge  $(v, u)$  in  $G(L)$ ; notice that  $(v, u)$  is an
      edge in  $G(L)$ , while  $(u, v)$  only exists in  $G_f$ , since
      it is a backward edge of  $G_f$  */

```

Lemma 10. If f is a flow in $G(L)$ at some iteration of Algorithm 2 and J is a $s \rightarrow t$ journey in G_f then $f' = \text{augment}(f, J)$ is also a flow in $G(L)$.

Proof. To show that $f' = \text{augment}(f, J)$ is a flow, since the node capacities are infinite, we need only show that it satisfies the edge capacity constraint, and that flow only moves via journeys, i.e., it satisfies the *time-respecting constraint*:

Capacity constraint If $(u, v) \in J$ is a forward edge, then $f'((u, v), l_{(u, v)}) = f((u, v), l_{(u, v)}) + b$ and $b \leq c_{uv} - f((u, v), l_{(u, v)})$. If $(v, u) \in J$ is a backward edge, then $f'((u, v), l_{(u, v)}) = f((u, v), l_{(u, v)}) - b$ and $b \leq f((u, v), l_{(u, v)})$.

In both cases, $0 \leq f'((u, v), l_{(u, v)}) \leq c_{uv}$.

Time-respecting constraint If all edges in J are forward, then the flow on all those edges is increased by b in $G(L)$ in a time-respecting way, i.e., flow units are being “pushed” through an edge $e = (u, v)$ at the time that e is available and only after they have reached the tail, u , of e . Suppose, now, that J has backward edges. Let $e = (u, v)$ be a backward edge in J and l_e be its label. Let $e' = (v, u)$ be the respective edge in $G(L)$. Note that $c_e = f(e', l_{e'})$ and $l_e = l_{e'}$. Also, note that for $e = (u, v)$ to be a backward edge, it must be that at a previous step of the Algorithm, some flow quantity was passing via another $s \rightarrow t$ journey, J_0 , through the edge $e' = (v, u)$ of $G(L)$; let e_0 be the edge of J_0 that is immediately before $e' = (v, u)$ in J_0 ; it must be $l_{e_0} < l_{e'} = l_e$ for J_0 to be a journey. Also, let (v, x) be the edge after e in J .

- Suppose that (v, x) is a forward edge. Clearly, $l_{(v, x)} > l_e = l_{e'}$; otherwise, J would not be a journey. Since e is an edge in J , after the augmentation b units of flow out of the c_e units that were previously being pushed through e' are now being pushed through (v, x) . We know that (v, x) becomes available *after* (u, v) (since J is a journey), so it becomes available also after (v, u) (since the label of (u, v) is the same as the label of (v, u)). Therefore, it becomes available also after e_0 (since $l_{e_0} < l_{e'} = l_e$). So, the time constraint is being satisfied.

- Suppose that (v, x) is a backward edge.

For (v, x) to be a backward edge, it must be that at a previous step of the Algorithm, some flow quantity was passing via another $s \rightarrow t$ journey, J'_0 , through the edge (x, v) of $G(L)$; let e'_0 be the edge of J'_0 that is immediately before (x, v) in J'_0 ; it must be $l_{e'_0} < l_{(x, v)} = l_{(v, x)}$ for J'_0 to be a journey. As in the previous case, the flow f' on both edges (v, u) and (x, v) in $G(L)$ is reduced by b after the augmentation. We direct b units of flow instead from x to y (respecting the time-constraint for the same reason as in the previous case), where (x, y) is the edge after (v, x) in J ; if that edge is also a backwards edge, the same logic applies until we find a subsequent edge in J that is a forward edge; such an edge will eventually be found because t is the last vertex in J , t has no outgoing edges in $G(L)$ and J is simple by construction, so the last edge in J must be forward. The b units of flow we direct to (x, y) go through it at time $l_{(x, y)}$, which is greater than $l_{(v, x)} = l_{(x, v)} > l_{e'_0}$, since (x, y) is after (v, x) in J . Therefore, the actual path that those b units follow from s to t is time-respecting.

□

Lemma 11. *At every step of Algorithm 2, the flow values $f((u, v), l_{(u, v)})$ (or $f((v, u), l_{(v, u)})$) and the residual capacities in G_f are integers.*

Proof. The initial flow and residual capacities are integers. Suppose the Lemma holds for j iterations of the algorithm. Then in the $(j + 1)^{th}$ iteration, the minimum edge capacity b is an integer and so the flow after the augmentation is an integer. □

Lemma 12. *Let f be a flow in $G(L)$ at some iteration of Algorithm 2 and f' be the flow after one augmentation. Then $v(f) < v(f')$.*

Proof. Let J be a $s \rightarrow t$ journey in G_f . The first edge, e , of J must leave s . The original network $G(L)$ has no incoming edges to s , so e must be a forward edge in G_f . J is simple by construction, so it never returns to s . Hence, the value of the flow increases by the flow on e . \square

Lemma 13. *Let $S = \sum_{e \in \delta_s^+} c_e$ be the sum of capacities of the outgoing edges of s . Algorithm 2 terminates after performing at most S augmentations, i.e., iterations of the **while** loop.*

Proof. The value of the flow increases by at least 1 at each iteration/augmentation. The maximum value of the flow cannot exceed S , so the Lemma holds. \square

Corollary 2. *The running time of Algorithm 2 is $O(Sm^2 \log m)$, where m is the number of edges, i.e., also time-edges, in $G(L)$.*

Proof. The following hold:

- The number of iterations of the Algorithm is at most S .
- The number of edges, and thus also time-edges, in G_f is at most $2m$.
- The time needed to find an augmenting journey, if one exists, is $O(m \log m)$ [3].

Therefore, the running time of Algorithm 2 is $O(Sm^2 \log m)$. \square

Lemma 14. *Given a single-labelled temporal flow network $(G(L) = (V, E, L), s, t, c)$ and a flow f in it, the following statements are equivalent:*

1. *There is a temporal cut C in $G(L)$ with $c(C) = v(f)$, where $c(C)$ is the capacity of the temporal cut, i.e., sum of capacities of the time-edges it contains.*
2. *f is a maximum temporal flow in $G(L)$.*
3. *There is no augmenting journey for f , i.e., there is no $s \rightarrow t$ journey in G_f .*

Proof. a) \Rightarrow b). Since for every flow f and any cut C , $v(f) \leq c(C)$ (that can be easily shown using the time-extended network), $c(C) = v(f)$ implies that f is maximum.

b) \Rightarrow c). If there is an augmenting journey for f , then $v(f)$ can be increased by increasing the flow along the time-edges of that journey, i.e., f is not maximum, which is a contradiction.

c) \Rightarrow a). Suppose that there is no augmenting journey for f . Consider the journey-decomposition of f and eliminate all flow from edges on journeys that never reach t ; let f' be the resulting flow. Clearly, $v(f') = v(f)$; also, there is no augmenting journey for f' iff there is no augmenting journey for f .

Now, consider $G_{f'}$ and let A be the set of nodes reachable via journeys from s in it; let $B = V \setminus A$. It is $s \in A$ and $t \in B$. Notice that all the time-edges in $G_{f'}$ that “cross” the partition must either have labels that are smaller than the ones on all the edges with both endpoints in A or have larger labels *but* be incoming to A ; the latter time-edges correspond to saturated time-edges in $G(L)$, i.e., their respective time-edges in $G(L)$ carry as much flow as their capacity; let T be the set of those saturated time-edges. Clearly, $v(f') = c(T)$, and thus $c(T) = v(f)$. It remains to show that T is a temporal cut in $G(L)$.

Assume to the contrary that T is not a temporal cut, i.e., that there is an $s \rightarrow t$ journey in $G(L)$ that uses *none* of the time-edges of T ; let $J = ((s, x_1, l_1), (x_1, x_2, l_2), \dots, (x_{\phi-1}, t, l_\phi))$ be that journey, for some $\phi \in \mathbb{N}^*$ and some nodes $x_i \in V$, $i = 1, \dots, \phi - 1$. Now, assume that $x_1 \notin A$ (in G_f); then, it must be that $(s, x_1, l_1) \in T$, which is a contradiction; therefore, it is $x_1 \in A$. Assume now that $x_2 \notin A$ (in G_f); then, it must be that $(x_1, x_2, l_2) \in T$, which is a contradiction; therefore, it is $x_2 \in A$. Following the same reasoning, we get $x_3, \dots, x_{\phi-1}, t \in A$. However, we already have that $t \in B = V \setminus A$, which contradicts the latter. Therefore, our assumption that T is not a temporal cut is false, and the implication holds. \square

Theorem 3. *The flow returned by Algorithm 2 is the maximum flow.*

Proof. For any flow f and any (minimal) temporal (s, t) cut \mathcal{C} , it holds $v(f) \leq c(\mathcal{C})$. For the flow f^* that the algorithm returns, there is some (minimal) temporal cut \mathcal{C}^* such that $v(f^*) = c(\mathcal{C}^*)$. If there was a flow f' with $v(f') > v(f^*)$, that would imply that $v(f') > c(\mathcal{C}^*)$, which contradicts the above, i.e., the fact that the value of any flow is upper bounded by the capacity of any minimal temporal (s, t) cut. Therefore, f^* is maximum. \square

5 Conclusions

We defined and studied here for the first time flows in temporal networks. We also considered random availabilities in some of the edges of our networks (mixed temporal networks). An interesting open problem is the existence of a FPTAS for the expected maximum flow value in mixed temporal networks.

References

- [1] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network Flows: Theory, Algorithms, and Applications*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1993.
- [2] Eleni C. Akrida, Leszek Gaŝieniec, George B. Mertzios, and Paul G. Spirakis. Ephemeral networks with random availability of links: Diameter and connectivity. In *Proceedings of the 26th ACM Symposium on Parallelism in Algorithms and Architectures (SPAA)*, 2014.
- [3] Eleni C. Akrida, Leszek Gaŝieniec, George B. Mertzios, and Paul G. Spirakis. On temporally connected graphs of small cost. In *Proceedings of the 13th Workshop on Approximation and Online Algorithms (WAOA)*, 2015.

- [4] Eleni C. Akrida and Paul G. Spirakis. On verifying and maintaining connectivity of interval temporal networks. In Prosenjit Bose, Leszek Antoni Gasieniec, Kay Römer, and Roger Wattenhofer, editors, *Algorithms for Sensor Systems - 11th International Symposium on Algorithms and Experiments for Wireless Sensor Networks, ALGOSENSORS 2015, Patras, Greece, September 17-18, 2015, Revised Selected Papers*, volume 9536 of *Lecture Notes in Computer Science*, pages 142–154. Springer, 2015. URL: http://dx.doi.org/10.1007/978-3-319-28472-9_11, doi:10.1007/978-3-319-28472-9_11.
- [5] Alexandr Andoni, Anupam Gupta, and Robert Krauthgamer. Towards $(1+\epsilon)$ -approximate flow sparsifiers. In Chandra Chekuri, editor, *Proceedings of the Twenty-Fifth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2014, Portland, Oregon, USA, January 5-7, 2014*, pages 279–293. SIAM, 2014. URL: <http://dx.doi.org/10.1137/1.9781611973402.20>, doi:10.1137/1.9781611973402.20.
- [6] J. E. Aronson. A survey of dynamic network flows. *Ann. Oper. Res.*, 20(1-4):1–66, August 1989. URL: <http://dx.doi.org/10.1007/BF02216922>, doi:10.1007/BF02216922.
- [7] C. Avin, M. Koucký, and Z. Lotker. How to explore a fast-changing world (cover time of a simple random walk on evolving graphs). In *Proceedings of the 35th International Colloquium on Automata, Languages and Programming (ICALP)*, pages 121–132, 2008.
- [8] Jatin Batra, Naveen Garg, Amit Kumar, Tobias Mömke, and Andreas Wiese. New approximation schemes for unsplittable flow on a path. In Piotr Indyk, editor, *Proceedings of the Twenty-Sixth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2015, San Diego, CA, USA, January 4-6, 2015*, pages 47–58. SIAM, 2015. URL: <http://dx.doi.org/10.1137/1.9781611973730.5>, doi:10.1137/1.9781611973730.5.
- [9] Nadine Baumann and Martin Skutella. Earliest arrival flows with multiple sources. *Math. Oper. Res.*, 34(2):499–512, 2009. URL: <http://dx.doi.org/10.1287/moor.1090.0382>, doi:10.1287/moor.1090.0382.
- [10] Kenneth A. Berman. Vulnerability of scheduled networks and a generalization of menger’s theorem. *Networks*, 28(3):125–134, 1996.
- [11] Arnaud Casteigts, Paola Flocchini, Walter Quattrociocchi, and Nicola Santoro. Time-varying graphs and dynamic networks. *International Journal of Parallel, Emergent and Distributed Systems (IJPEDS)*, 27(5):387–408, 2012.
- [12] Andrea E. F. Clementi, Claudio Macci, Angelo Monti, Francesco Pasquale, and Riccardo Silvestri. Flooding time of edge-markovian evolving graphs. *SIAM Journal on Discrete Mathematics (SIDMA)*, 24(4):1694–1712, 2010.
- [13] Chinmoy Dutta, Gopal Pandurangan, Rajmohan Rajaraman, Zhifeng Sun, and Emanuele Viola. On the complexity of information spreading in dynamic networks. In *Proceedings of the 24th Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 717–736, 2013.

- [14] Friedrich Eisenbrand, Andreas Karrenbauer, Martin Skutella, and Chihao Xu. Multiline addressing by network flow. *Algorithmica*, 53(4):583–596, 2009. URL: <http://dx.doi.org/10.1007/s00453-008-9252-5>, doi:10.1007/s00453-008-9252-5.
- [15] Khaled M. Elbassioni, Naveen Garg, Divya Gupta, Amit Kumar, Vishal Narula, and Arindam Pal. Approximation algorithms for the unsplit-table flow problem on paths and trees. In Deepak D’Souza, Telikepalli Kavitha, and Jaikumar Radhakrishnan, editors, *IARCS Annual Conference on Foundations of Software Technology and Theoretical Computer Science, FSTTCS 2012, December 15-17, 2012, Hyderabad, India*, volume 18 of *LIPIcs*, pages 267–275. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2012. URL: <http://dx.doi.org/10.4230/LIPIcs.FSTTCS.2012.267>, doi:10.4230/LIPIcs.FSTTCS.2012.267.
- [16] Thomas Erlebach, Michael Hoffmann, and Frank Kammer. On temporal graph exploration. In Magnús M. Halldórsson, Kazuo Iwama, Naoki Kobayashi, and Bettina Speckmann, editors, *Automata, Languages, and Programming - 42nd International Colloquium, ICALP 2015, Kyoto, Japan, July 6-10, 2015, Proceedings, Part I*, volume 9134 of *Lecture Notes in Computer Science*, pages 444–455. Springer, 2015. URL: http://dx.doi.org/10.1007/978-3-662-47672-7_36, doi:10.1007/978-3-662-47672-7_36.
- [17] Lisa Fleischer and Martin Skutella. Quickest flows over time. *SIAM J. Comput.*, 36(6):1600–1630, 2007. URL: <http://dx.doi.org/10.1137/S0097539703427215>, doi:10.1137/S0097539703427215.
- [18] Lisa Fleischer and Éva Tardos. Efficient continuous-time dynamic network flow algorithms. *Operations Research Letters*, 23(3-5):71–80, 1998.
- [19] D. R. Ford and D. R. Fulkerson. *Flows in Networks*. Princeton University Press, Princeton, NJ, USA, 2010.
- [20] Dimitris Fotakis, Spyros C. Kontogiannis, Elias Koutsoupias, Marios Mavronicolas, and Paul G. Spirakis. The structure and complexity of nash equilibria for a selfish routing game. *Theor. Comput. Sci.*, 410(36):3305–3326, 2009. URL: <http://dx.doi.org/10.1016/j.tcs.2008.01.004>, doi:10.1016/j.tcs.2008.01.004.
- [21] Mohsen Ghaffari, Andreas Karrenbauer, Fabian Kuhn, Christoph Lenzen, and Boaz Patt-Shamir. Near-optimal distributed maximum flow: Extended abstract. In Chryssis Georgiou and Paul G. Spirakis, editors, *Proceedings of the 2015 ACM Symposium on Principles of Distributed Computing, PODC 2015, Donostia-San Sebastián, Spain, July 21 - 23, 2015*, pages 81–90. ACM, 2015. URL: <http://doi.acm.org/10.1145/2767386.2767440>, doi:10.1145/2767386.2767440.
- [22] Martin Groß and Martin Skutella. Generalized maximum flows over time. In *Approximation and Online Algorithms - 9th International Workshop, WAOA 2011, Saarbrücken, Germany, September 8-9, 2011, Revised Selected Papers*, pages 247–260, 2011. URL: http://dx.doi.org/10.1007/978-3-642-29116-6_21, doi:10.1007/978-3-642-29116-6_21.

- [23] Mohammad Taghi Hajiaghayi and Harald Räcke. An $o(\sqrt{n})$ -approximation algorithm for directed sparsest cut. *Inf. Process. Lett.*, 97(4):156–160, 2006. URL: <http://dx.doi.org/10.1016/j.ipl.2005.10.005>, doi: 10.1016/j.ipl.2005.10.005.
- [24] Bruce Hoppe and Éva Tardos. The quickest transshipment problem. *Math. Oper. Res.*, 25(1):36–62, 2000. URL: <http://dx.doi.org/10.1287/moor.25.1.36.15211>, doi:10.1287/moor.25.1.36.15211.
- [25] Bruce Edward Hoppe. Phd thesis: Efficient dynamic network flow algorithms, 1995.
- [26] Jan-Philipp W. Kappmeier. *Generalizations of Flows over Time with Applications in Evacuation Optimization*. epubli GmbH, 2015. URL: <http://www.epubli.de/shop/isbn/9783737532389>.
- [27] David Kempe, Jon M. Kleinberg, and Amit Kumar. Connectivity and inference problems for temporal networks. In *Proceedings of the 32nd annual ACM symposium on Theory of computing (STOC)*, pages 504–513, 2000.
- [28] Jon M. Kleinberg, Yuval Rabani, and Éva Tardos. Allocating bandwidth for bursty connections. *SIAM J. Comput.*, 30(1):191–217, 2000. URL: <http://dx.doi.org/10.1137/S0097539797329142>, doi:10.1137/S0097539797329142.
- [29] Bettina Klinz and Gerhard J. Woeginger. One, two, three, many, or: complexity aspects of dynamic network flows with dedicated arcs. *Oper. Res. Lett.*, 22(4-5):119–127, 1998. URL: [http://dx.doi.org/10.1016/S0167-6377\(98\)00009-1](http://dx.doi.org/10.1016/S0167-6377(98)00009-1), doi:10.1016/S0167-6377(98)00009-1.
- [30] Ronald Koch, Ebrahim Nasrabadi, and Martin Skutella. Continuous and discrete flows over time - A general model based on measure theory. *Math. Meth. of OR*, 73(3):301–337, 2011. URL: <http://dx.doi.org/10.1007/s00186-011-0357-2>, doi:10.1007/s00186-011-0357-2.
- [31] Jakub Lacki, Yahav Nussbaum, Piotr Sankowski, and Christian Wulff-Nilsen. Single source - all sinks max flows in planar digraphs. In *53rd Annual IEEE Symposium on Foundations of Computer Science, FOCS 2012, New Brunswick, NJ, USA, October 20-23, 2012*, pages 599–608. IEEE Computer Society, 2012. URL: <http://dx.doi.org/10.1109/FOCS.2012.66>, doi:10.1109/FOCS.2012.66.
- [32] Aleksander Madry. Fast approximation algorithms for cut-based problems in undirected graphs. In *51th Annual IEEE Symposium on Foundations of Computer Science, FOCS 2010, October 23-26, 2010, Las Vegas, Nevada, USA*, pages 245–254. IEEE Computer Society, 2010. URL: <http://dx.doi.org/10.1109/FOCS.2010.30>, doi:10.1109/FOCS.2010.30.
- [33] George B. Mertzios, Othon Michail, Ioannis Chatzigiannakis, and Paul G. Spirakis. Temporal network optimization subject to connectivity constraints. In *Proceedings of the 40th International Colloquium on Automata, Languages and Programming (ICALP), Part II*, pages 657–668, 2013.

- [34] Othon Michail and Paul G. Spirakis. Traveling salesman problems in temporal graphs. In Erzsébet Csuha-J-Varij, Martin Dietzfelbinger, and Zoltán Ésik, editors, *Mathematical Foundations of Computer Science 2014 - 39th International Symposium, MFCS 2014, Budapest, Hungary, August 25-29, 2014. Proceedings, Part II*, volume 8635 of *Lecture Notes in Computer Science*, pages 553–564. Springer, 2014. URL: http://dx.doi.org/10.1007/978-3-662-44465-8_47, doi:10.1007/978-3-662-44465-8_47.
- [35] Vincenzo Nicosia, John Tang, Cecilia Mascolo, Mirco Musolesi, Giovanni Russo, and Vito Latora. Graph Metrics for Temporal Networks. In Peter Holme and Jari Saramäki, editors, *Temporal Networks, Understanding Complex Systems*, pages 15–40. Springer, May 2013.
- [36] Regina O’Dell and Roger Wattenhofer. Information dissemination in highly dynamic graphs. In *Proceedings of the 2005 joint workshop on Foundations of Mobile Computing (DIALM-POMC)*, pages 104–110, 2005.
- [37] James B. Orlin. Max flows in $o(nm)$ time, or better. In Dan Boneh, Tim Roughgarden, and Joan Feigenbaum, editors, *Symposium on Theory of Computing Conference, STOC’13, Palo Alto, CA, USA, June 1-4, 2013*, pages 765–774. ACM, 2013. URL: <http://doi.acm.org/10.1145/2488608.2488705>, doi:10.1145/2488608.2488705.
- [38] Christos M. Papadimitriou. *Computational complexity*. Addison-Wesley, Reading, Massachusetts, 1994.
- [39] Warren B Powell, Patrick Jaillet, and Amedeo Odoni. Stochastic and dynamic networks and routing. *Handbooks in operations research and management science*, 8:141–295, 1995.
- [40] Christian Scheideler. Models and techniques for communication in dynamic networks. In *Proceedings of the 19th Annual Symposium on Theoretical Aspects of Computer Science (STACS)*, volume 2285, pages 27–49, 2002.
- [41] Martin Skutella. An introduction to network flows over time. In *Research Trends in Combinatorial Optimization, Bonn Workshop on Combinatorial Optimization, November 3-7, 2008, Bonn, Germany*, pages 451–482, 2008. URL: http://dx.doi.org/10.1007/978-3-540-76796-1_21, doi:10.1007/978-3-540-76796-1_21.